BAB 2
LANDASAN TEORI

2.1 Game

Game merupakan suatu sistem yang memiliki aturan-aturan tertentu dimana pemain akan terlibat di dalam suatu permasalahan sehingga dapat menghasilkan suatu hasil yang dapat diukur yaitu menang atau kalah[19]. *Game* merupakan sesuatu hal yang dimainkan dengan suatu aturan tertentu yang biasa digunakan untuk tujuan kesenangan dan dapat juga digunakan untuk tujuan pendidikan. Untuk memahami lebih dalam lagi tentang *game,* perlu dipahami terlebih dahulu konsepnya. Secara konsep, teori *game* adalah sebagai berikut[19] :

1. *Number of Players*

   Hampir semua jenis permainan papan yang memiliki sistem pencarian langkah berbasis algoritma pada AI hanya memiliki dua pemain. Sebagian besar bentuk dasar dari algoritma-algoritma tersebut hanya terbatas untuk dua pemain.

2. *Plies, Move and Turns*

   Suatu hal umum dalam teori permainan adalah giliran (*turns)* seorang pemain sebagai suatu lapisan (*ply*) didalam suatu permainan dan pemain yang melakukan gilirannya dalam satu putaran disebut langkah (*move*).

3. *The Goal of the Game*

   Tujuan umum permainan berbasis strategi adalah untuk mendapatkan kemenangan. Sebagai pemain, pemain menang jika semua lawan pemain kalah. Hal ini dikenal sebagai permainan zero-sum, yaitu kemenangan pemain adalah kekalahan pemain lain. Jika pemain mencetak 1 poin untuk menang, maka akan setara dengan mencetak -1 poin untuk kalah. Untuk kasus permainan non-zero-sum, semua bisa menang atau semua bisa kalah, pemain hanya akan fokus pada kemenangan.

4. *Information*

   Dalam permainan papan seperti catur, Checkers, Go, dan Reversi, kedua pemain mengetahui segala sesuatu tentang

kondisi dalam permainan. Pemain mengetahui hasil dari setiap gerakan yang dilakukan dan pilihan yang akan dilakukan pemain untuk langkah berikutnya. Pemain mengetahui semua ini dari awal permainan. Jenis permainan ini disebut "informasi yang sempurna".

2.1.1 Action-RPG

Action-Role Playing Game atau ARPG adalah bentuk sebuah subgenre yang didefinisikan dari *role-playing video game* yang menggabungkan unsur-unsur dari tindakan atau action-adventure game, menekankan tindakan real-time di mana pemain memiliki kontrol langsung pada karakter, bukan turn-based atau berbasis menu tempur. Permainan ini sering menggunakan sistem tempur yang mirip dengan hack dan slash atau game shooter.

2.1.2 NPC

Penjabaran NPC (Non Player Character) adalah karakter game yang dikendalikan oleh sistem bukan oleh pemain. NPC (Non Player Character) melayani sejumlah tujuan di dalam game, termasuk :

1. Sebagai perangkat alur : NPC dapat digunakan untuk memajukan alur cerita
2. Untuk bantuan atau musuh : NPC dapat bertindak sebagai mitra untuk pemain atau dapat menjadi musuh dalam game
3. Fungsi game : NPC sering melayani sebagai menyimpan poin, toko item, poin kesehatan regenerasi dan sebagainya.

2.1.3 Story Board

Storyboard adalah visualisasi ide dari aplikasi yang akan dibangun, sehingga dapat memberikan gambaran dari aplikasi yang akan dihasilkan. Storyboard dapat dikatakan juga *visual script* yang akan dijadikan outline dari sebuah proyek, ditampilkan *shot by shot* yang biasa disebut dengan istilah *scene*.

Storyboard sekarang lebih banyak digunakan untuk membuat kerangka pembuatan websites dan proyek media interaktif lainnya seperti iklan, film

pendek, games, media pembelajaran interaktif ketika dalam tahap perancangan /desain.

2.1.4 Gameplay

Gameplay adalah cara sebuah gamer untuk berinteraksi dengan game tertentu. Gameplay dalam Video Game sangatlah berpengaruh dalam Pembuatan Game. semakin Game tersebut mudah untuk dimainkan maka semakin banyak yang menyukainya walaupun kadang kala ada beberapa game yang dimana Gameplay nya tidak mudah akan tetapi karena penjelasan tutorial yang mudah dimengerti maka game tersebut juga dapat banyak peminatnya.

2.2 Logika Fuzzy

Logika *fuzzy* adalah cabang dari sistem kecerdasan buatan (*Artificial Intelegent*) yang meniru kemampuan manusia dalam berfikir ke dalam bentuk algoritma yang kemudian dijalankan oleh mesin. Algoritma ini digunakan dalam berbagai aplikasi pemrosesan data yang tidak dapat direpresentasikan dalam bentuk biner. Logika fuzzy menginterpretasikan statemen yang samar menjadi sebuah pengertian yang logis.

Logika *Fuzzy* pertama kali diperkenalkan oleh *Prof. Lotfi Zadeh* seorang kebangsaan Iran yang menjadi guru besar di *University of California at Berkeley* pada tahun 1965 dalam papernya yang monumental. Dalam paper tersebut dipaparkan ide dasar *fuzzy set* yang meliputi *inclusion, union, intersection, complement, relation* dan *convexity*. Pelopor aplikasi *fuzzy set* dalam bidang kontrol, yang merupakan aplikasi pertama dan utama dari *fuzzy set* adalah *Prof. Ebrahim Mamdani* dan kawan-kawan dari *Queen Mary College London*. Penerapan kontrol *fuzzy* secara nyata di industri banyak dipelopori para ahli dari Jepang, misalnya *Prof. Sugeno* dari *Tokyo Institute of Technology*, *Prof.Yamakawa* dari *Kyusu Institute of Technology*, *Togay* dan *Watanabe* dari *Bell Telephone Labs* [3]. Komponen - komponen fuzzy sebagai berikut :

- Himpunan Fuzzy

  Himpunan *fuzzy* merupakan suatu pengembangan lebih lanjut tentang konsep himpunan dalam matematika. Himpunan *Fuzzy*

adalah rentang nilai-nilai. Masing-masing nilai mempunyai derajat keanggotaan (*membership*) antara 0 sampai dengan 1. Ungkapan logika *Boolean* menggambarkan nilai-nilai "benar" atau "salah". Logika *fuzzy* menggunakan ungkapan misalnya : "sangat lambat","agak sedang", "sangat cepat" dan lain-lain untuk mengungkapkan derajat intensitasnya [8].

- Fuzzifikasi

  Proses fuzzifikasi merupakan proses untuk mengubah variabel *non fuzzy* (variabel numerik) menjadi variabel *fuzzy* (variabel linguistik). Nilai masukan-masukan yang masih dalam bentuk variabel numerik yang telah dikuantisasi sebelum diolah oleh pengendali *fuzzy* harus diubah terlebih dahulu ke dalam variabel *fuzzy*. Melalui fungsi keanggotaan yang telah disusun maka nilai-nilai masukan tersebut menjadi informasi *fuzzy* yang berguna nantinya untuk proses pengolahan secara *fuzzy* pula. Proses ini disebut fuzzifikasi [8].

- Inferencing (Rule Base)

  Pada umumnya, aturan-aturan *fuzzy* dinyatakan dalam bentuk *"IF…THEN"* yang merupakan inti dari relasi *fuzzy*. Relasi *fuzzy*, dinyatakan dengan *R*, juga disebut implikasi *fuzzy* [8]. Untuk mendapatkan aturan *"IF…..THEN"* ada dua cara utama :

  1. Menanyakan ke operator manusia yang dengan cara manual telah mampu mengendalikan sistem tersebut, dikenal dengan *"human expert".*

     Dengan menggunakan algoritma pelatihan berdasarkan data-data masukan dan keluaran.

- Defuzzifikasi

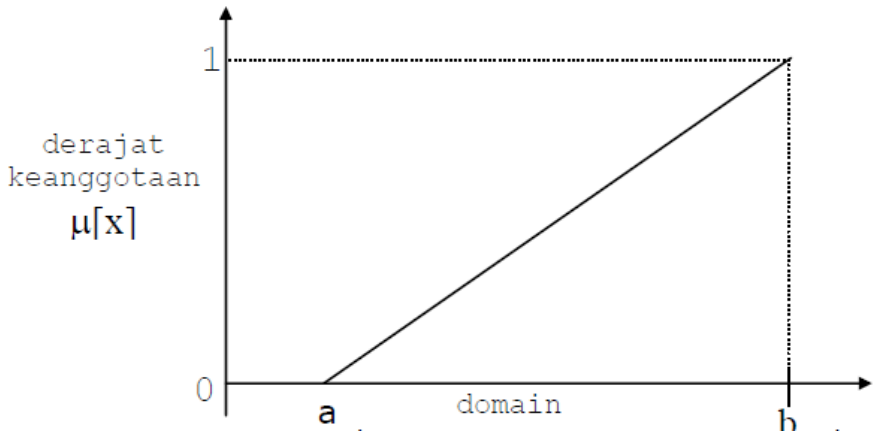
  Keputusan yang dihasilkan dari proses penalaran masih dalam bentuk *fuzzy*, yaitu berupa derajat keanggotaan keluaran. Hasil ini harus diubah kembali menjadi variabel numerik *non fuzzy* melalui proses defuzzifikasi [8].

2.2.1 Fungsi Keanggotaan

Fungsi Keanggotaan (membership function) adalah suatu kurva yang menunjukkan pemetaan titik-titik input data ke dalam nilai keanggotaannya (sering juga disebut dengan derajat keanggotaan) yang memiliki interval antara 0 sampai 1. Salah satu cara yang dapat digunakan untuk mendapatkan nilai keanggotaan adalah dengan melalui pendekatan fungsi. Ada beberapa fungsi yang bisa digunakan.

a. Representasi Linear

Pada representasi linear, pemetaan input ke derajat keanggotannya digambarkan sebagai suatu garis lurus. Bentuk ini paling sederhana dan menjadi pilihan yang baik untuk mendekati suatu konsep yang kurang jelas. Ada 2 keadaan himpunan fuzzy yang linear. Pertama, kenaikan himpunan dimulai pada nilai domain yang memiliki derajat keanggotaan nol (0) bergerak ke kanan menuju ke nilai domain yang memiliki derajat keanggotaan lebih tinggi. Grafik ditunjukkan pada gambar berikut

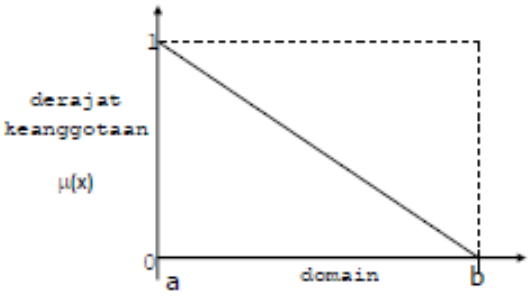


Gambar 2.1 Representasi Linear Naik

Fungsi Keanggotaan:

$$\mu[x] = \begin{cases} 0; & x \le a \\ (x-a)/(b-a); & a \le x \le b \\ 1; & x \ge b \end{cases} \quad (2.1)$$

Kedua, representasi linear turun. Garis lurus dimulai dari nilai domain dengan derajar keanggotaan tertinggi pada sisi kiri, kemudian bergerak menurun ke nilai domain yang memiliki derajat keanggotaan lebih rendah. Grafik ditunjukkan pada gambar berikut :



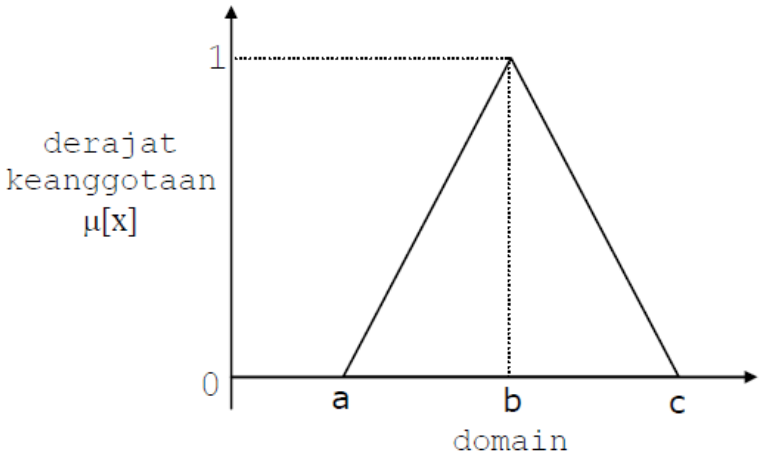
Gambar 2.2 Representasi Linear Turun

Fungsi Keanggotaan :

$$\mu(x) = \begin{cases} (b - x) / (b - a) ; & a \leq x \leq b \\ 0; & x \geq b \end{cases} \quad (2.2)$$

b. Representasi kurva segitiga

Kurva Segitiga pada dasarnya merupakan gabungan antara 2 garis (linear) seperti terlihat pada Gambar dibawah ini [8]



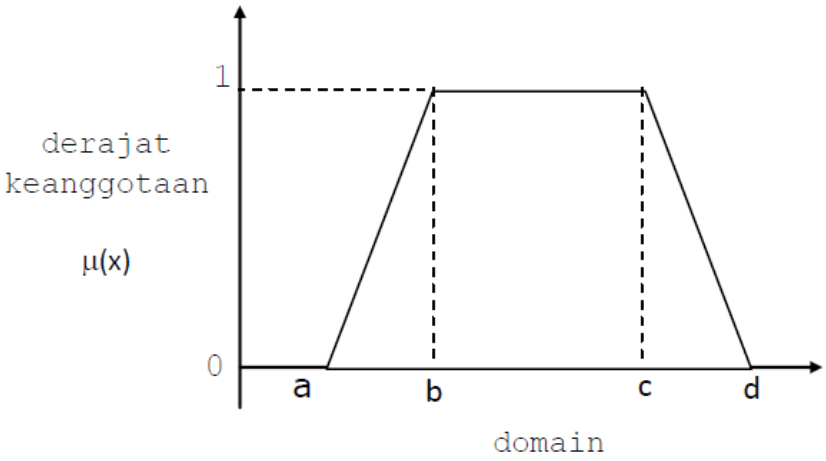
Gambar 2.3 Representasi kurva segitiga

Fungsi Keanggotaan:

$$\mu[x] = \begin{cases} 0; & x \le a \text{ atau } x \ge c \\ (x - a)/(b - a); & a \le x \le b \\ (b - x)/(c - b); & b \le x \le c \end{cases} \quad (2.3)$$

c. Repsentasi Kurva Trapesium

Kurva trapesium pada dasarnya seperti bentuk segitiga, hanya saja ada beberapa titik yang memuliki nilai keanggotaan 1. Berikut grafik representasi kurva trapesium



Gambar 2.4 Representasi Kurva Trapesium

Fungsi Keanggotaan :

$$\mu(x) = \begin{cases} 0; & x \le a \text{ atau } x \ge d \\ (x - a)/(b - a); & a \le x \le b \\ 1; & b \le x \le c \\ (d - x)/(d - c); & c \le x \le d \end{cases} \quad (2.4)$$

2.2.2 Metode Tsukamoto

Pada dasarnya metode tsukamoto mengaplikasikan penalaran monoton pada setiap aturannya. Kalau pada penalaran monoton, sistem hanya memiliki satu aturan, pada metode tsukamoto sistem terdiri dari beberapa aturan. Karena menggunakan konsep dasar penalaran monoton, pada metode tsukamoto setiap konsekuen pada aturan yang berbentuk IF-THEN harus direpresentasikan dengan suatu himpunan fuzzy dengan fungsi keanggotaan yang monoton. Output hasil inferensi dari tiap – tiap aturan diberikan secara tegas (crisp) berdasarkan α-predikat (fire strength). Proses agregasi antar aturan dilakukan, dan hasil akhirnya diperoleh dengan menggunakan defuzzy dengan konsep rata – rata terbobot. Misalkan ada variabel input, yaitu x dan y, serta satu variabel output yaitu z. Variabel x terbagi atas 2 himpunan yaitu A1 dan A2, variabel y terbagi atas 2 himpunan juga, yaitu B1 dan B2, sedangkan variabel output z terbagi atas 2 himpunan yaitu C1 dan C2 harus merupakan himpunan yang bersifat monoton. Diberikan 2 aturan sebagai berikut :

1. IF x is A1 dan y is B2 THEN z is C1
2. IF x is A2 dan Y is B2 THEN z is C1

2.2.3 Metode Mamdani

salah satu metode yang memiliki struktur yang sederhana, metode ini palin sering digunakan dalam aplikasi – aplikasi. Metode ini menggunakan Operasi MIN dan MAX atau MAX PRODUCT. Dalam pembuatan aplikasi yang menggunakan metode mamdani ini, dapat menggunakan 4 langkah tahapan untuk mendapatkan ouutput yang diharapkan. Tahapan – tahapan tersebut diantaranya :

1. Fuzzifikasi
2. Pembantukan basis Pengetahuan Fuzzy (RULE dalam bentuk IF THEN)
3. Aplikasi fungsi implikasi menggunakan fungsi MIN dan Komposisi antar-rule menggunakan fungsi MAX (menghasilkan himpunan fuzzy baru)
4. Defuzzifikasi menggunakan “metode Centroid”

   Input dari proses defuzzifikasi adalah suatu himpunan fuzzy yang diperoleh dari komposisi auran – aturan fuzzy, sedangkan output yang dihasilkan merupakan suatu bilangan pada domain himpunan fuzzy tersebut

2.2.2 Metode Sugeno

Penalaran dengan metode sugeno hampir sama dengan penalaran mamdani, hanya saja output (konsekuen) sistem tidak berupa himpunan fuzzy, melainkan berupa konstanta atau persamaan linear. Metode ini diperkenalkan oleh Takagi-Sugeno Kang pada tahun 1985, sehingga metode ini sering juga dinamakan dengan metode TSK.

Proses fuzzy inference dapat dibagi dalam 5 bagian, yaitu :

1. Fuzzyfikasi Input : FIS mengambil masukan – masukan dan menentukan derajat keanggotaanya dalam semua fuzzy set
2. Operasi logika fuzzy : hasil dari operasi ini adalah derajat kebenaran antecedent yang berupa bilangan tunggal
3. Impilkasi : merupakan proses mendapatkan consequent atau keluaran sebuah IF-THEN rule berdasarkan derajat kebenaran antacedent. Proses ini menggunakan mengamabil nilai MIN dari 2 bilangan
4. Agregasi : yaitu proses mengkombinasikan semua keluaran IF-THEN rule menjadi sebuah fuzzy set tunggal menggunakan fungsi MAX
5. Defuzzyfikasi : keluaran defuzzyfikasi adalah bilangan tunggal. Cara mendapatkannya ada beberapa versi, yaitu *centroid, bisector, middle of maximum, largest of maximum* dan *smallest of maximum.*

Secara umum model fuzzy Sugeno terdiri dari 2 jenis:

1. Model fuzzy Sugeno orde-nol : IF input1 = x dan input2 = y THEN maka z = k
2. Model fuzzy Sugeno orde-satu : IF input1 = x dan input2 = y THEN maka z = ax+by+c

Defuzzyfikasi dilakukan dengan cara mencari rata-ratanya (weight average) :

$$\text{Final Output} = \frac{\sum_{i=1}^{n} \text{WiZi}}{\sum_{i=1}^{n} \text{Wi}}$$

Dimana :

N = Jumlah Fuzzy Rule

W = Bobot Hasil Implikasi

Z = output perilaku musuh

2.3 Algoritma Floyd-Warshall

a. Pengertian Algoritma Floyd-Warshall

Secara umum algoritma *Floyd-Warshall* merupakan salah satu varian dari pemrograman dinamis, yaitu suatu metode yang melakukan pemecahan masalah dengan memandang solusi yang akan diperoleh sebagai suatu keputusan yang saling terkait. Artinya solusi-solusi tersebut dibentuk dari solusi yang berasal dari tahap sebelumnya dan ada kemungkinan solusi lebih dari satu.

Algoritma yang ditemukan oleh warshall untuk mencari *path* terpendek merupakan algoritma yang sederhana dan mudah implementasikan. Masukan algoritma Warshall adalah matriks hubung graf dan keluarannya adalah path terpendek dari semua titik. Dalam usaha untuk mencari path terpendek, algoritma warshall memulai iterasi dari titik awalnya kemudian memperpanjang *path* dengan mengevaluasi titik demi titik hingga mencapai titik tujuan dengan jumlah bobot seminimum mungkin [18].

Algoritma Floyd Warshall memiliki input graf berbobot (V,E) yang berupa daftar titik (nodes/verteks V) dan daftar sisi (edge E). Jumlah bobot sisi-sisi pada suatu lintasan adalah bobot sisi tersebut. Sisi pada E diperbolehkan memiliki bobot negatif, akan tetapi tidak diperbolehkan bagi graf ini untuk memiliki siklus dengan bobot negatif. Algoritma ini menghitung bobot terkecil dari semua sisi yang menghubungkan sebuah pasangan titik. Prinsip yang dipegang oleh algoritma Floyd-Warshall adalah prinsip optimalitas, yaitu jika solusi tetap optimal, maka bagian solusi sampai suatu tahap (misalnya tahap ke i) juga optimal.

Menurut Siang, sebagaimana dikutip oleh Sani *et al.* (2013 : 3), algoritma Floyd Warshall untuk mencari lintasan terpendek. Dimisalkan W(0) adalah matrik ketetanggaan awal graf berarah berbobot. W* adalah matrik ketetanggaan berbobot terkecil dengan sama dengan W_{ij} * lintasan terpendek dari titik V_i ke V_j

1) $W = W_0$
2) Untuk k = 1 hingga n, lakukan :
   Untuk i = 1 hingga n, lakukan :

Untuk j = 1 hingga n, lakukan :

Jika W [i, j] > W [i, k] + W [k, j] maka

Tukar W [i, j] dengan W [i, k] + W [k, j]

3) W* = W

Keterangan:

W = matrik

W_0 = matrik ketetanggaan awal

k = iterasi 1 sampai ke-n

i = titik awal v_i

j = titik awal v_j

W* = hasil matrik setelah perbandingan

b. Algoritma Floyd-Warshall Untuk Graf Tidak Berarah

Algoritma Floyd-Warshall dikembangkan oleh R. W. Floyd sehingga algoritma floyd warshall dapat digunakan untuk mencari jarak antara semua titik graf. Algoritma ini sangat efisien dari sudut pandang penyimpanan data karena dapat diimplementasikan dengan hanya pengubahan sebuah matriks jarak.

Algoritma Floyd-Warshall memiliki input graf tak berarah dan berbobot (V, E) yang berupa himpunan titik (titik V) dan himpunan sisi (sisi E). Bobot sisi e dapat diberi simbol d(i,j).\

Diketahui n titik dalam graf tidak berarah adalah v_1, v_2, v_3, ... v_n untuk menemukan lintasan terpendek diantara semua pasangan titik, dengan langkah sebagai berikut :

1. Untuk $i \neq j$, jika v_i v_j adalah sisi, ambil d(i,j) sebagai bobot dari sisi tersebut. Jika tidak ada sisi yang menghubungkan langsung antara i dan j ditulis $d(i,j) = \infty$, untuk i=j maka ditulis d(i,j) = 0
2. Untuk k = 1 sampai n

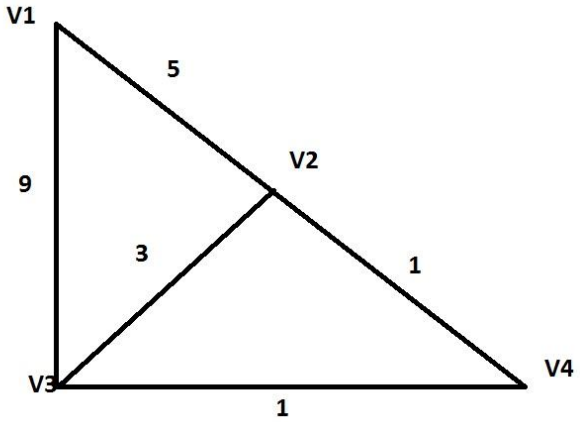
   Untuk i,j = 1 sampai n ditulis d(i,j) = min { + d(k,j)}

   Nilai akhir dari d(i,j) adalah jarak dari vi ke vj. (Goodaire & Parmenter, 1998:382)

Dari prosedur diatas dapat dilihat bahwa iterasi ke-k ($1 \leq k \leq n$). Mula-mula algoritma untuk jarak dari v_i ke v_j adalah panjang dari sisi v_i ke v_j adalah

panjang sisi v_iv_j. Setelah iterasi pertama pada langkah 2 (k=1), jarak yang diperoleh digantikan dengan panjang dari lintasan v_iv1v_j. Setelah iterasi k pada algoritma ini dapat ditentukan jarak terpendek dari v_i ke v_j pada titik-titik, v1, v2,...,vk, jarak adalah setelah iterasi ke k=n dengan mengambil d(i,j) sebagai lintasan terpendek dari i ke j.

Contoh perhitungan Algoritma Floyd Warshall sebagai berikut :



Gambar 2.5 Contoh Graf Berbobot

Bentuk Matriknya sebagai berikut :

Tabel 2.1 Matrik W(0)

	V1	V2	V3	V4
V1	0	5	9	∞
V2	5	0	3	1
V3	9	3	0	1
V4	∞	1	1	0

K=1

$$d(1,1) = \min \{d(1,1), d(1,1) + d(1,1)\} = \min\{0, 0 + 0\} = 0$$

d(1,2) = min {d(1,2), d(1,1) + d(1,2)} = min{5, 0 + 5} = 5
d(1,3) = min {d(1,3), d(1,1) + d(1,3)} = min{9, 0 + 9} = 9
d(1,4) = min {d(1,4), d(1,1) + d(1,4)} = min{∞, 0 + ∞} = ∞
d(2,1) = min {d(2,1), d(2,1) + d(1,1)} = min{5, 5 + 0} = 5
d(2,2) = min {d(2,2), d(2,1) + d(1,2)} = min{0, 5 + 5} = 0
d(2,3) = min {d(2,3), d(2,1) + d(1,3)} = min{3, 5 + 9} = 3
d(2,4) = min {d(2,4), d(2,1) + d(1,4)} = min{1, 5 + ∞} = 1
d(3,1) = min {d(3,1), d(3,1) + d(1,1)} = min{9, 9 + 0} = 9
d(3,2) = min {d(3,2), d(3,1) + d(1,2)} = min{3, 9 + 5} = 3
d(3,3) = min {d(3,3), d(3,1) + d(1,3)} = min{0, 9 + 9} = 0
d(3,4) = min {d(3,4), d(3,1) + d(1,4)} = min{1, 9 + ∞} = 1
d(4,1) = min {d(4,1), d(4,1) + d(1,1)} = min{∞, ∞ + 0} = ∞
d(4,2) = min {d(4,2), d(4,1) + d(1,2)} = min{1, ∞ + 5} = 1
d(4,3) = min {d(4,3), d(4,1) + d(1,3)} = min{1, ∞ + 9} = 1
d(4,4) = min {d(4,4), d(4,1) + d(1,4)} = min{0, ∞ + ∞} = 0

Sehingga diperoleh matriknya sebagai berikut :

Tabel 2.2 W(1)

	V1	V2	V3	V4
V1	0	5	9	∞
V2	5	0	3	1
V3	9	3	0	1
V4	∞	1	1	0

K = 2

d(1,1) = min {d(1,1), d(1,2) + d(2,1)} = min{0, 5 + 5} = 0
d(1,2) = min {d(1,2), d(1,2) + d(2,2)} = min{5, 5 + 0} = 5
d(1,3) = min {d(1,3), d(1,2) + d(2,3)} = min{9, 5 + 3} = 8

d(1,4) = min {d(1,4), d(1,2) + d(2,4)} = min{∞, 5 + 1} = 6
d(2,1) = min {d(2,1), d(2,2) + d(2,1)} = min{5, 0 + 5} = 5
d(2,2) = min {d(2,2), d(2,2) + d(2,2)} = min{0, 0 + 0} = 0
d(2,3) = min {d(2,3), d(2,2) + d(2,3)} = min{3, 0 + 3} = 3
d(2,4) = min {d(2,4), d(2,2) + d(2,4)} = min{1, 0 + 1) = 1
d(3,1) = min {d(3,1), d(3,2) + d(2,1)} = min{9, 3 + 5} = 8
d(3,2) = min {d(3,2), d(3,2) + d(2,2)} = min{3, 3 + 0} = 3
d(3,3) = min {d(3,3), d(3,2) + d(2,3)} = min{0, 3 + 3} = 0
d(3,4) = min {d(3,4), d(3,2) + d(2,4)} = min{1, 3 + 1} = 1
d(4,1) = min {d(4,1), d(4,2) + d(2,1)} = min{∞, 1 + 5} = 6
d(4,2) = min {d(4,2), d(4,2) + d(2,2)} = min{1, 1 + 0} = 1
d(4,3) = min {d(4,3), d(4,2) + d(2,3)} = min{1, 1 + 3} = 1
d(4,4) = min {d(4,4), d(4,2) + d(2,4)} = min{0, 1 + 1} = 0

Sehingga diperoleh matriknya sebagai berikut :

Tabel 2.3 Matrik W(2)

	V1	V2	V3	V4
V1	0	5	8	6
V2	5	0	3	1
V3	8	3	0	1
V4	6	1	1	0

K = 3

d(1,1) = min {d(1,1), d(1,3) + d(3,1)} = min{0, 8 + 8} = 0
d(1,2) = min {d(1,2), d(1,3) + d(3,2)} = min{5, 8 + 3} = 5
d(1,3) = min {d(1,3), d(1,3) + d(3,3)} = min{8, 8 + 0} = 8
d(1,4) = min {d(1,4), d(1,3) + d(3,4)} = min{9, 8 + 1} = 9
d(2,1) = min {d(2,1), d(2,3) + d(3,1)} = min{5, 3 + 8} = 5

d(2,2) = min {d(2,2), d(2,3) + d(3,2)} = min{0, 3 + 3} = 0
d(2,3) = min {d(2,3), d(2,3) + d(3,3)} = min{3, 3 + 0} = 3
d(2,4) = min {d(2,4), d(2,3) + d(3,4)} = min{1, 3 + 1) = 1
d(3,1) = min {d(3,1), d(3,3) + d(3,1)} = min{8, 0 + 8} = 8
d(3,2) = min {d(3,2), d(3,3) + d(3,2)} = min{3, 0 + 3} = 3
d(3,3) = min {d(3,3), d(3,3) + d(3,3)} = min{0, 0 + 0} = 0
d(3,4) = min {d(3,4), d(3,3) + d(3,4)} = min{1, 0 + 1} = 1
d(4,1) = min {d(4,1), d(4,3) + d(3,1)} = min{6, 1 + 8} = 6
d(4,2) = min {d(4,2), d(4,3) + d(3,2)} = min{1, 1 + 3} = 1
d(4,3) = min {d(4,3), d(4,3) + d(3,3)} = min{1, 1 + 0} = 1
d(4,4) = min {d(4,4), d(4,3) + d(3,4)} = min{0, 1 + 1} = 0

Sehingga diperoleh matriknya sebagai berikut

Tabel 2.4 Matrik W(3)

	V1	V2	V3	V4
V1	0	5	8	6
V2	5	0	3	1
V3	8	3	0	1
V4	6	1	1	0

K = 4

d(1,1) = min {d(1,1), d(1,4) + d(4,1)} = min{0, 6 + 5} = 0
d(1,2) = min {d(1,2), d(1,4) + d(4,2)} = min{5, 6 + 1} = 5
d(1,3) = min {d(1,3), d(1,4) + d(4,3)} = min{8, 6 + 1} = 7
d(1,4) = min {d(1,4), d(1,4) + d(4,4)} = min{6, 6 + 0} = 6
d(2,1) = min {d(2,1), d(2,4) + d(4,1)} = min{5, 1 + 6} = 5
d(2,2) = min {d(2,2), d(2,4) + d(4,2)} = min{0, 1 + 1} = 0
d(2,3) = min {d(2,3), d(2,4) + d(4,3)} = min{3, 1 + 1} = 2

d(2,4) = min {d(2,4), d(2,4) + d(4,4)} = min{1, 1 + 1) = 1
d(3,1) = min {d(3,1), d(3,4) + d(4,1)} = min{8, 1 + 6} = 7
d(3,2) = min {d(3,2), d(3,4) + d(4,2)} = min{3, 1 + 1} = 2
d(3,3) = min {d(3,3), d(3,4) + d(4,3)} = min{0, 1 + 1} = 0
d(3,4) = min {d(3,4), d(3,4) + d(4,4)} = min{1, 1 + 0} = 1
d(4,1) = min {d(4,1), d(4,4) + d(4,1)} = min{6, 0 + 6} = 6
d(4,2) = min {d(4,2), d(4,4) + d(4,2)} = min{1, 0 + 1} = 1
d(4,3) = min {d(4,3), d(4,4) + d(4,3)} = min{1, 0 + 1} = 1
d(4,4) = min {d(4,4), d(4,4) + d(4,4)} = min{0, 0 + 0} = 0

Sehingga diperoleh matriknya sebagai berikut :

Tabel 2.5 Matrik W(4)

	V1	V2	V3	V4
V1	0	5	7	6
V2	5	0	2	1
V3	7	2	0	1
V4	6	1	1	0

Sehingga diperoleh suatu lintasan terpendek pada setiap titiknya dari matrik di atas dapat disimpulkan bahwa jarak dari V_1 titik ke V_1 adalah 0, jarak dari titik V_1 ke V_2 adalah 5, V_1 ke V_3 adalah 7, V_1 ke V_4 adalah 6 dan sebagainya.

2.4 Android

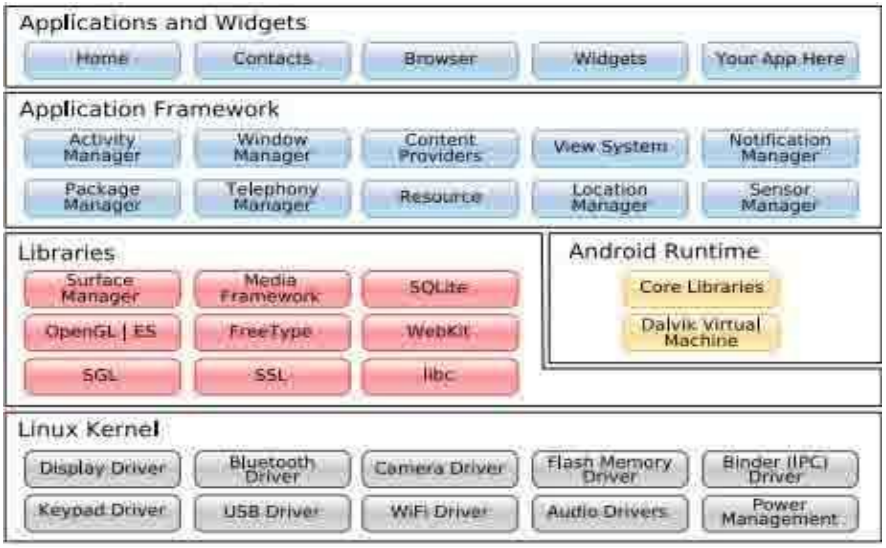
Android adalah sistem operasi berbasis Linux yang dipergunakan sebagai pengelola sumber daya perangkat keras, baik untuk ponsel, smartphone dan juga PC tablet. Secara umum Android adalah platform yang terbuka (Open Source) bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh berbagai piranti bergerak. Pada saat perilisan perdana, 5 November 2007, Android bersama *Open Handset Alliance* menyatakan

mendukung pengembangan standar terbuka pada perangkat seluler. Di lain pihak, Google merilis kode – kode Android di bawah lisensi Apache. Di dunia ini terdapat dua jenis distributor sistem operasi Android. Pertama yang mendapat dukungan penuh dari Google atau *Google Mail Services* (GMS) dan kedua adalah yang benar–benar bebas distribusinya tanpa dukungan langsung Google atau dikenal sebagai *Open Handset Distribution* (OHD)

Gambar 2.6 Android

2.4.1 Anatomi Aplikasi Android

Pada zaman sekarang ini Android memang sudah dikenal oleh hampir semua kalangan masyarakat di dunia, terlebih lagi di Indonesia. Bahkan sudah menjadi suatu kebutuhan yang tidak dapat terpisahkan dari keseharian kita. Sistem Operasi atau yang sering dikenal dengan istilah Operating System (OS) pada Android ini sangatlah unik dan mampu memberi kemudahan pada penggunanya, itulah salah satu faktor mengapa Android menjadi pilihan tiada duanya. Sistem operasi Android terdiri dari beberapa unsur seperti yang ada pada gambar 2.0 [11]. Secara sederhana arsitektur Android merupakan sebuah kernel Linux dan sekumpulan pustaka C/C++ dalam suatu *framework* yang menyediakan dan mengatur alur proses aplikasi.



Gambar 2.7 Arsitektur Android

2.4.2 Application Layer

Layer pertama pada OS Android, biasa dinamakan layer Applications. Layer ini biasa digunakan yang berhubungan dengan aplikasi inti yang berjalan pada Android OS. Lapisan aplikasi merupakan lapisan yang paling tampak pada pengguna ketika menjalankan program. Pengguna hanya akan melihat program ketika digunakan tanpa mengetahui proses yang terjadi dibalik lapisan aplikasi. Lapisan ini berjalan dalam Android *runtime* dengan menggunakan kelas dan servis yang tersedia pada *framework* aplikasi.

Pada Android semua aplikasi, baik aplikasi inti (*native*) maupun aplikasi pihak ketiga berjalan diatas lapisan aplikasi dengan menggunakan pustaka API (*Application Programming Interface*) yang sama [5].

2.4.3 Application Framework

Application Framework merupakan layer dimana pembuat aplikasi menggunakan komponen-komponen yang ada di sini untuk membuat aplikasi mereka. Kerangka Aplikasi menyediakan kelas-kelas yang dapat digunakan untuk mengembangkan aplikasi Android. Selain itu juga menyediakan abstraksi generik untuk mengakses perangkat, serta mengatur tampilan *user interface* dan sumber daya aplikasi. Menurut [7] ada beberapa Bagian terpenting dalam kerangka aplikasi Android adalah sebagai berikut:

- *Activity Manager*, berfungsi untuk mengontrol siklus hidup aplikasi danmenjaga keadaan *Backstack* untuk navigasi penggunaan.
- *Content Providers*, berfungsi untuk merangkum data yang memungkinkandigunakan oleh aplikasi lainnya dan seperti daftar nama.
- *Resuource Manager*, untuk mengatur sumber daya yang ada dalamprogram. Serta menyediakan akses sumber daya diluar kode program, seperti karakter, grafik dan file layout.
- *Location Manager*, berfungsi untuk memberikan informasi detailmengenai lokasi perangkat Android berada.
- *Notification Manager*, mencakup berbagai macam peringatan seperti, pesan masuk, janji dan lain sebagainya yang akan ditampilkan pada *statusbar*

2.4.4 Android Runtime

Ada yang membedakan Android jika di bandingkan dengan sstem operasi lain yang juga mengimplementasikan Linux. Android *Runtime* merupakan mesin virtual yang membuat aplikasi Android menjadi lebih tangguh dengan paket pustaka yang telah ada. Terdapat 2 bagian dalam android *Rontime*, yaitu:

- Pustaka Inti, Android dikembangkan melalui bahasa pemrograman Java, tapi Android *Runtime* bukanlah mesin virtual Java. Pustaka inti Android menyediakan hampir semua fungsi yang terdapat pada pustaka Java serta beberapa pustakakhusus Android.
- Mesin Virtual Dalvik merupakan sebuah mesin virtual yangdikembangkan oleh Dan Bornstein yang terinspirasi dari nama sebuahperkampungan yang berada di Iceland [7]. *Dalvik* hanyalah interpreter mesinvirtual yang mengeksekusi *file* dalam format *Dalvik Executable* (*.dex). Dengan format ini Dalvik akan mengoptimalkan efisiensi penyimpanandan pengalamatan memori pada *file* yang dieksekusi. Dalvik

berjalan diatas *kernel Linux* 2.6, dengan fungsi dasar seperti *threading* dan manajemen memori yang terbatas [11]

2.4.5 Libraries

Android menggunakan beberapa paket pustaka yang terdapat pada C/C++ dengan standar *Berkeley Software Distribution* (BSD) hanya setengah dari yang aslinya untuk tertanam pada kernel Linux. Beberapa pustaka diantaranya:

- Media Library untuk memutar dan merekam berbagai macam format audio dan video.
- *Surface Manager* untuk mengatur hak akses layer dari berbagai aplikasi.
- *Graphic Library* termasuk didalamnya *SGL* dan *OpenGL*, untuk tampilan2D dan 3D.
- *SQLite* untuk mengatur relasi database yang digunakan pada aplikasi.
- *SSl* dan*WebKit* untuk browser dan keamanan internet.

Pustaka-pustaka tersebut bukanlah aplikasi yang berjalan sendiri, namun dapat digunakan oleh program yang berada di level atasnya. Sejak versi Android 1.5, pengembang dapat membuat dan menggunakan pustaka sendiri menggunakan *Native Development Toolkit* (NDK).

2.5 Unified Modelling Languange (UML)

Unified Modeling Language **(UML)** merupakan sistem arsitektur yang bekerja dalam OOAD (*Object-Oriented Analysis/Design*) dengan satu bahasa yang konsisten untuk menentukan, visualisasi, mengkontruksi, dan mendokumentasikan *artifact* (sepotong informasi yang digunakan atau dihasilkan dalam suatu proses rekayasa software, dapat berupa model, deskripsi, atau software) yang terdapat dalam sistem software. UML merupakan bahasa pemodelan yang paling sukses dari tiga metode OO yang telah ada sebelumnya, yaitu Booch, OMT (*Object Modeling Technique*), dan OOSE (*Object-Oriented Software Engineering*).

UML mendefinisikan diagram-diagram sebagai berikut:

1. *Use case diagram*
2. *Activity diagram*
3. *Sequence diagram*
4. *Class diagram*

UML dapat digunakan untuk membuat model untuk semua jenis perangkat lunak, dimana perangkat lunak tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan class dan operation dalam konsep dasarnya, maka akan lebih cocok untuk penulisan piranti lunak dalam bahasa berorientasi objek.

2.5.1 *Use Case Diagram*

Use Case atau diagram *use case* merupakan pemodelan yang digunakan untuk menggambarkan kelakuan (*behavior)* dari sistem yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut.

Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case*.

1. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem yang akan dibuat diluar sistem yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
2. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesar antarunit atau aktor.

2.5.2 *Activity Diagram*

Activity diagram menggambarkan berbagai alur aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alur berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

Activity diagram merupakan *state diagram* khusus, dimana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan *behaviour* internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari *level* atas secara umum[20].

Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut [6]:

1. Rancangan proses bisnis di mana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokan tampilan dari sistem/*user interface* di mana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
3. Rancangan pengunjian di mana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.

2.5.3 *Sequence Diagram*

Diagram sekuen atau *sequence diagram* adalah diagram yang menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antarobjek. Oleh karena itu untuk menggambarkan diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Banyaknya diagram sekuen yang harus digambarkan adalah sebanyak pendefinisian *use case* yang memiliki prose situ sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak.

2.5.4 *Class Diagram*

Class diagram merupakan diagram statis. *Class diagram* mewakili pandangan statis dari aplikasi. *Class diagram* tidak hanya digunakan untuk

memvisualisasikan, menggambarkan, dan mendokumentasikan perbedaan aspek dari sistem, tetapi juga untuk membangun kode *executable* dari aplikasi *software*.

Class diagram menampilkan koleksi dari *class*, *interface*, *associations*, *collaborations*, dan *constraints*. Hal ini juga dikenal sebagai diagram struktural. *Class diagram* memberikan pandangan dari kelas desain dari sistem. *Class diagram* mengandung representasi dari kelas dan paket dan bagaimana mereka berhubungan.

Berikut merupakan komponen-komponen lain yang terdapat pada *class diagram:*

1. *Main class* yaitu kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.
2. *Interface class* merupakan kelas yang mendefinisikan dan mengatur tampilan ke pemakai. Biasanya juga disebut kelas *boundaries*.
3. Kelas yang diambil dari pendefinisian *usecase* merupakan kelas yang menangani fungsi-fungsi yang harus ada dan diambil dari pendefinisian *usecase*.
4. Kelas Entitas merupakan kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data.

Dalam suatu diagram kelas *atribut* dan *method* dapat memiliki salah satu sifat berikut :

a. *Private* : tidak dapat dipanggil dari luar kelas yang bersangkutan
b. *Protected* : Hanya dapat dipanggil oleh kelas yang bersangkutan dan anak-anak kelas yang mewarisinya.
c. *Public* : Dapat dipanggil oleh siapa saja.

2.6 C Sharp (C#)

C# (dibaca: C sharp) merupakan sebuah bahasa pemrograman yang berorientasi objek yang dikembangkan oleh Microsoft sebagai bagian dari inisiatif kerangka .NET Framework. Bahasa pemrograman ini dibuat berbasiskan bahasa C++ yang telah dipengaruhi oleh aspek-aspek ataupun fitur bahasa yang terdapat pada bahasa-bahasa pemrograman lainnya seperti Java, Delphi, Visual Basic, dan lain-lain) dengan beberapa penyederhanaan [8].

2.7 Unity

Unity 3D merupakan sebuah game engine yang dibuat Unity Technology oleh David Helgason di tahun 2004. Kelebihan Unity dibandingkan dengan *game engine* lainnya adalah untuk membuat game cross platform. Dengan Unity 3D, game yang dibuat dapat dimainkan di berbagai perangkat, seperti smartphone dan game console. Unity itu sendiri dapat membuat game seperti RPG, shooter, racing, dan lain sebagainya.

Untuk meningkatkan kualitas pemetaan atau tokoh dalam game, Unity 3D mendukung penggunaan software pengolah gambar lain, seperti Blender, Adobe Photoshop, Adobe Fireworks. Unity juga memiliki IDE yaitu MonoDevelop yang bertujuan untuk mengintegrasikan semua script yang dibuat kedalam unity sehingga dapat langsung diproses. Scripting yang digunakan pada unity dibangun menggunakan Mono 2.6. Mono 2.6 merupakan implementasi open source dari .Net Framework. Bahasa pemrograman yang didukung Unity 3D antara lain JavaScript, C#, dan Boo (menggunakan sintaks Phyton).

Unity juga menyertakan server unity asset, sebuah solusi terkontrol untuk developer game asset dan script. Server tersebut menggunakan PostgreSQL sebagai backend, sistem audio dibuat menggunakan FMOD library (dengan kemampuan untuk memutar Ogg Vorbis compressed audio), video playback menggunakan Theora codec, engine daratan dan vegetasi (dimana mensupport tree billboarding, Occlusion Culling dengan Umbra), built-in lightmapping dan global illumination dengan Beast, multiplayer networking menggunakan RakNet, dan navigasi mesh pencari jalur built-in.

Pada November 2010, Unity meluncurkan Unity Asset Store yaitu sebuah resource yang hadir di Unity Editor. Asset Store terdiri dari koleksi lebih dari 4400 asset packages, beserta 3D models, textures dan materials, sistem particle, musik dan efek suara, tutorial dan project, scripting package, editor extensions dan servis online.

Selain itu, unity juga memiliki support built-in untuk PhysX physics engine dari Nvidia dengan penambahan kemampuan untuk simulasi real-time cloth pada arbitary dan skinned meshes, thick ray cast, dan collision layers.

Gambar 2.8 Logo Unity

2.7.1 Prosedur penggunaan

Ketika Unity pertama kali dibuka biasanya akan menampilkan sample project. Untuk memulai project baru jalankan perintah File -> New Project. Maka akan muncul window Unity Project Wizard. Pada window itu anda akan diminta memasukkan lokasi dimana project anda, dan Package apa saja yang anda butuhkan pada project anda. Jika anda ragu untuk memilih Package yang akan diimport, maka lebih baik anda tidak mengimport package sama sekali.